PHENIX analysis framework (a.k.a Fun4All)

Jin Huang (BNL)
Many slides borrowed from Chris Pinkenburg (BNL)

PHENIX 2014 SpinFest at UIUC
July 28, 2PM

PHENIX

Overview

Raw data
- Format: *"Phenix Raw Data Format"* (PRDF)
- Size: 3PB (Run14), ~10PB (All runs)
- Storage: Cache disk (short term) and HPSS (permanent) in RCF

Data production
- Input: PRDF, Calibration
- Output: nDST (special formatted ROOT file for PHENIX)
- Output size ~1/3 of PRDF, stored in cache disks in RCF

Analysis train (a.k.a. Taxi)
- Input: nDST, recalibrations, (small) user inputs
- Output: Histograms, smaller DSTs (a.k.a. pDST) and/or ROOT files contains your TObjects

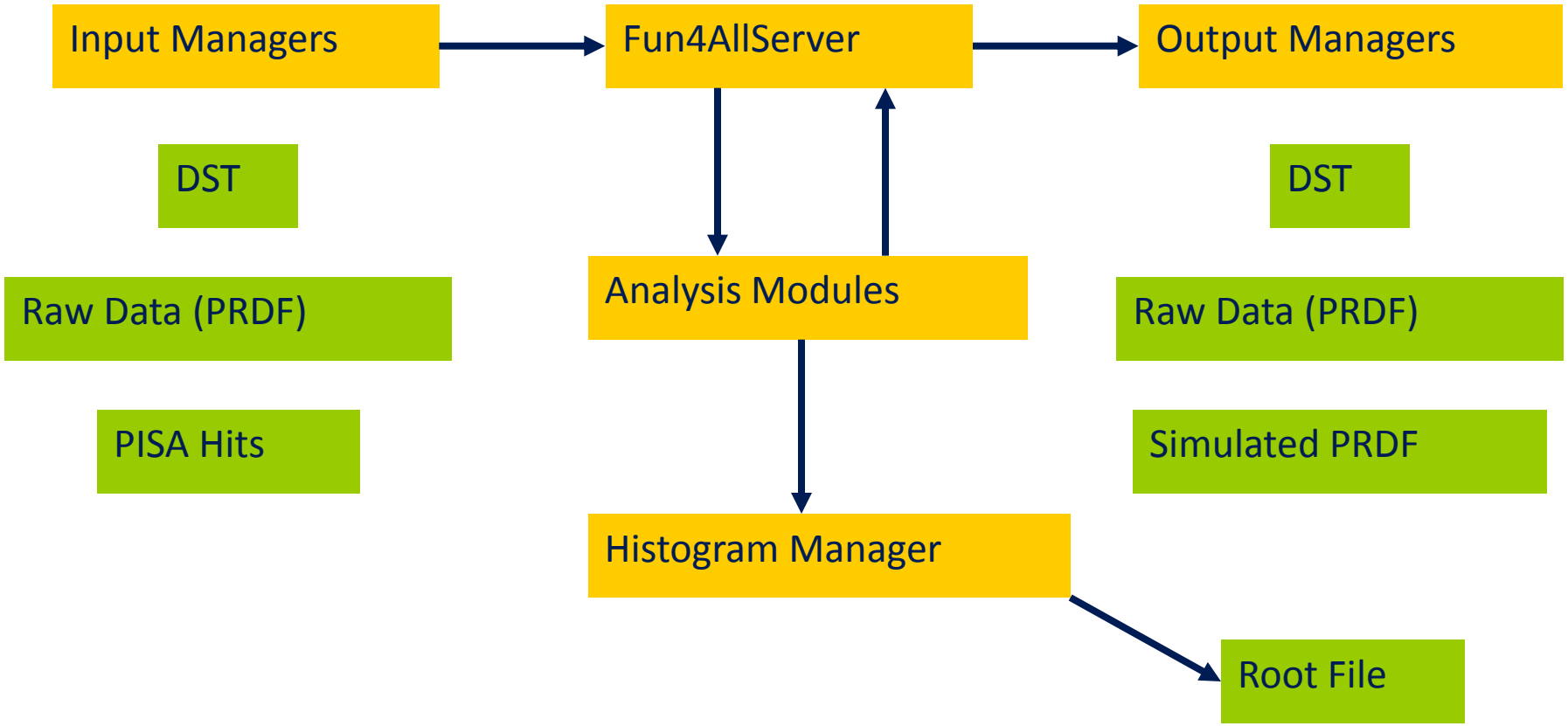
Final analysis
- Signal characterization, background fraction
- Form asymmetry
- Physics plots

Under PHENIX analysis frame a.k.a. Fun4All

Simulation also supported in the same framework using same reco code as data!

From Dr. Chris Pinkenburg (BNL)

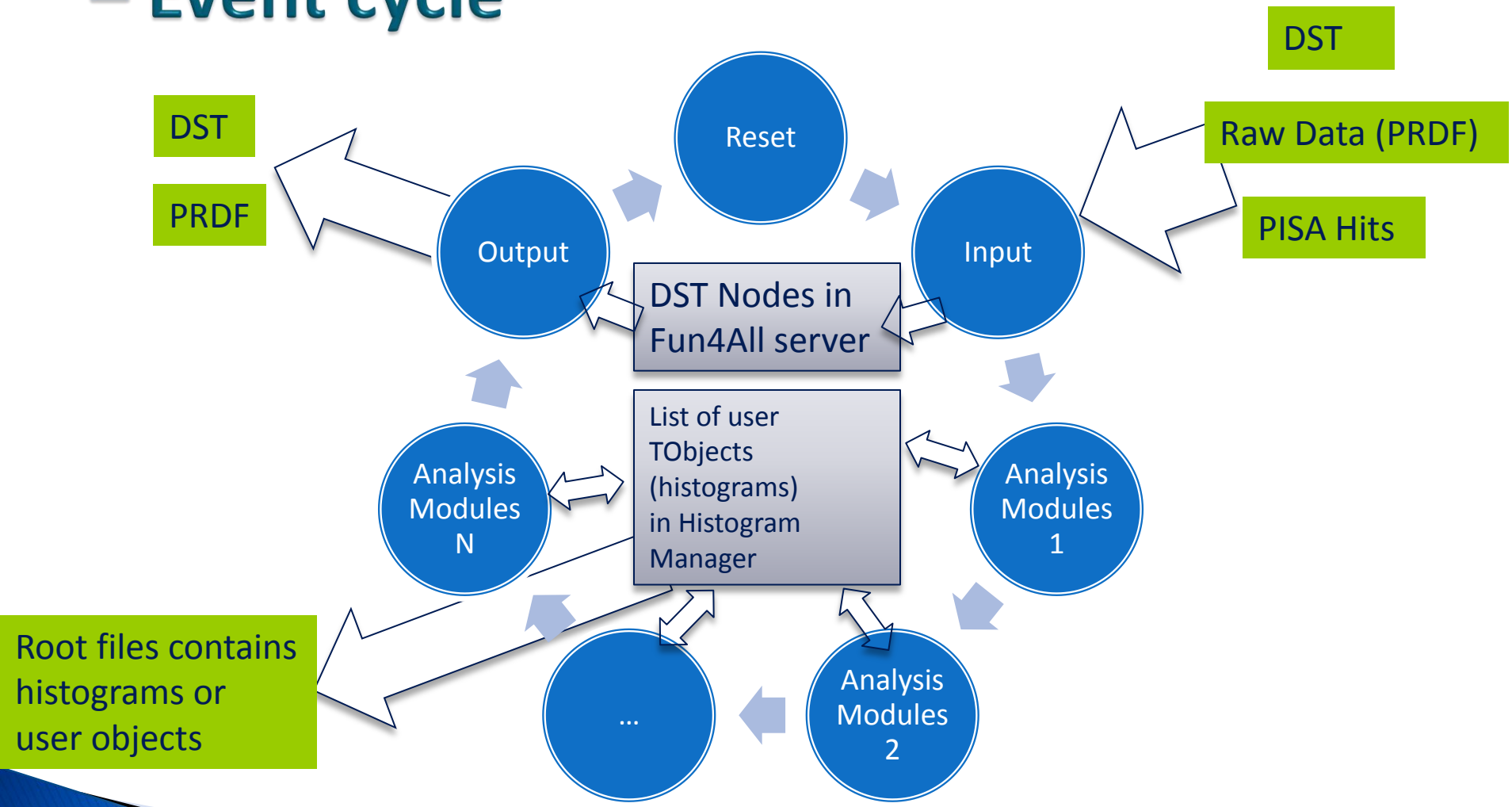
Structure of Fun4All – Overall flow



From Dr. Chris Pinkenburg (BNL)

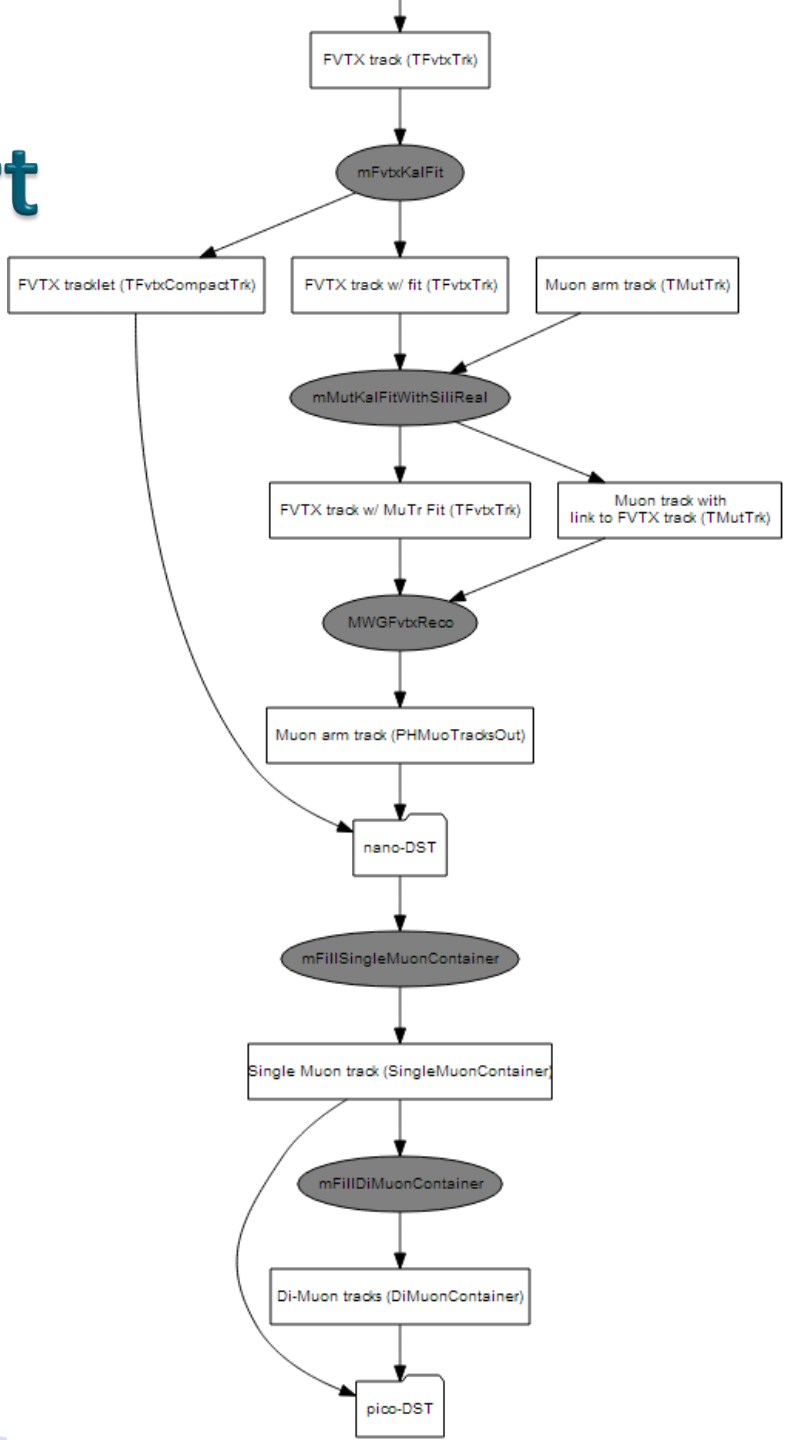
That's all there is to it (as long as you don't look under the hood)

Structure of Fun4All – Event cycle



One real life flow chart

- Example of FVTX reco chart
- Available interactively at http://www.phenix.bnl.gov/WWW/offline/doxygen/html/d9/d96/FVTX.html#Data_flow



Node Tree

- The Node Tree is at the center of the Phenix software universe (but it's more or less invisible to you). It's the way we organize our data.
- We have 3 different Types of Nodes:
  - PHCompositeNode: contains other Nodes
  - PHDataNode: contains any object
  - PHIODataNode: contains objects which can be written out
  - PHCompositeNodes and PHIODataNodes can be written to a file and read back
- DST: contains root trees, the node structure is reflected by the branch names
  - We currently save 2 root trees in each output file, one which contains the eventwise information, one which contains the runwise information
- Input Managers put Data on the node tree, output managers save selected nodes to a file.
- Node tree will be print out every time you run Fun4All server

Node Tree for real

- TOP (PHCompositeNode)/
  - DCM (PHCompositeNode)/
  - DST (PHCompositeNode)/
    - PhHadCglList (PHIODataNode)
    - EventHeader (PHIODataNode)
    - Sync (PHIODataNode)
    - TrigLvl1 (PHIODataNode)
    - PHGlobal (PHIODataNode)
    - emcClusterContainer (PHIODataNode)
    - AccCluster (PHIODataNode)
    - PHCentralTrack (PHIODataNode)
    - ReactionPlaneObject (PHIODataNode)
  - RUN (PHCompositeNode)/
    - RunHeader (PHIODataNode)
    - TrigRunLvl1 (PHIODataNode)
    - TrigRunLvl2 (PHIODataNode)
    - Flags (PHIODataNode)
    - DetectorGeometry (PHIODataNode)
  - PAR (PHCompositeNode)/
    - PRDF (PHDataNode)

Fun4All prints it out after everything is said and done in the BeginRun(), this is the tree You see when running our analysis train

These Nodes are create by default

These Nodes are “special” – they serve as default for the I/O, the objects under the DST Node are reset after every event to prevent mixing of events. You can select objects under the DST Node for saving, objects under the RUN Node are all saved in the output file

Fun4All can keep multiple node trees and Input/Output Managers can override their default Node where to put the data, but then things get too complicated for this occasion This is only needed for special applications which are not mainstream yet (e.g.embedding).

It's all about choices - Input

- **Fun4AllDstInputManager**: Reads DST's, if reading multiple input files it makes sure all data originates from the same event
- **Fun4AllNoSyncDstInputManager**: Reads DST's but doesn't check events for consistency (needed when reading simulated input together with real data for embedding or if you just feel like screwing up)
- **Fun4AllPisaInputManager**: Reads PISA Hits files
- **Fun4AllPrdfInputManager**: Raw Data (PRDF) uses

It's all about choices - Output

- **Fun4AllDstOutputManager**: Write DST's
- **Fun4AllEventOutputManager**: Writes Events in prdf format (packet selection possible)
- **Fun4AllPrdfOutputManager**: Write simulated raw data file.

Caveat:
You can only write Events if the input are Events (PRDF File)

Your Analysis Module

You need to inherit from the SubsysReco Baseclass (offline/framework/fun4all/SubsysReco.h) which gives the methods which are called by Fun4All. If you don't implement all of them it's perfectly fine (the beauty of base classes)

In your class you can implement as many additional methods
As you like, but these are the ones called by Fun4All

- Init(PHCompositeNode *topNode): called once at startup
- InitRun(PHCompositeNode *topNode): called whenever data from a new run is encountered
- Process_event (PHCompositeNode *topNode): called for every event
- ResetEvent(PHCompositeNode *topNode): called after each event is processed so you can clean up leftovers of this event in your code
- EndRun(const int runnumber): called before the InitRun is called (caveat the Node tree already contains the data from the first event of the new run)
- End(PHCompositeNode *topNode): Last call before we quit

Histogram OutPut

Use the Fun4AllHistoManager to safely save histograms and other TObject of yours to root files. Protected against problems in ROOT memory management.

```
#include “Fun4AllHistoManager.h”
MyAnalysis::MyAnalysis()
{
  HistoManager = new Fun4AllHistoManager("FUN4EXAMPLE Analyzer");
  HistoManager->setOutfileName( “myhistos.root”);
}
MyAnalysys::Init(PHCompositeNode *topNode)
{
  myhist1 = new TH1F(….); // myhist1 should be declared in MyAnalysis.h
  HistoManager->registerHistogram(myhist1);
}
```

This will save your histograms when executing the Fun4AllServer::End() in the file “myhistos.root”. If you don’t set the filename it will construct a name from the name of your analysis module

Analysis train (a.k.a. Taxi)

Official instruction: **offline WIKI / Analysis_Train**. Current coordinator: **Amaresh Datta** (UNM)

- The portal to get PHENIX data (few PB) for your analysis
  - Only way in PHENIX to be guaranteed to get all events from a given data set
  - Optimizing for computing resource (disk access/CPU time, etc.)
- You request to run providing a module and macro:
  - Create analysis module(s) → CVS:offline/AnalysisTrain
  - Macro to run the taxi → CVS:offline/AnalysisTrain/pat/macro
  - Check reconstruction against code-checkers
- Automated gatekeeper validity the code and arrange it to run as soon as possible
- Your defined output files (histograms, DSTs, ROOT objects) saved to your PWG disk (/spin1 or /spin2)

Code checkers

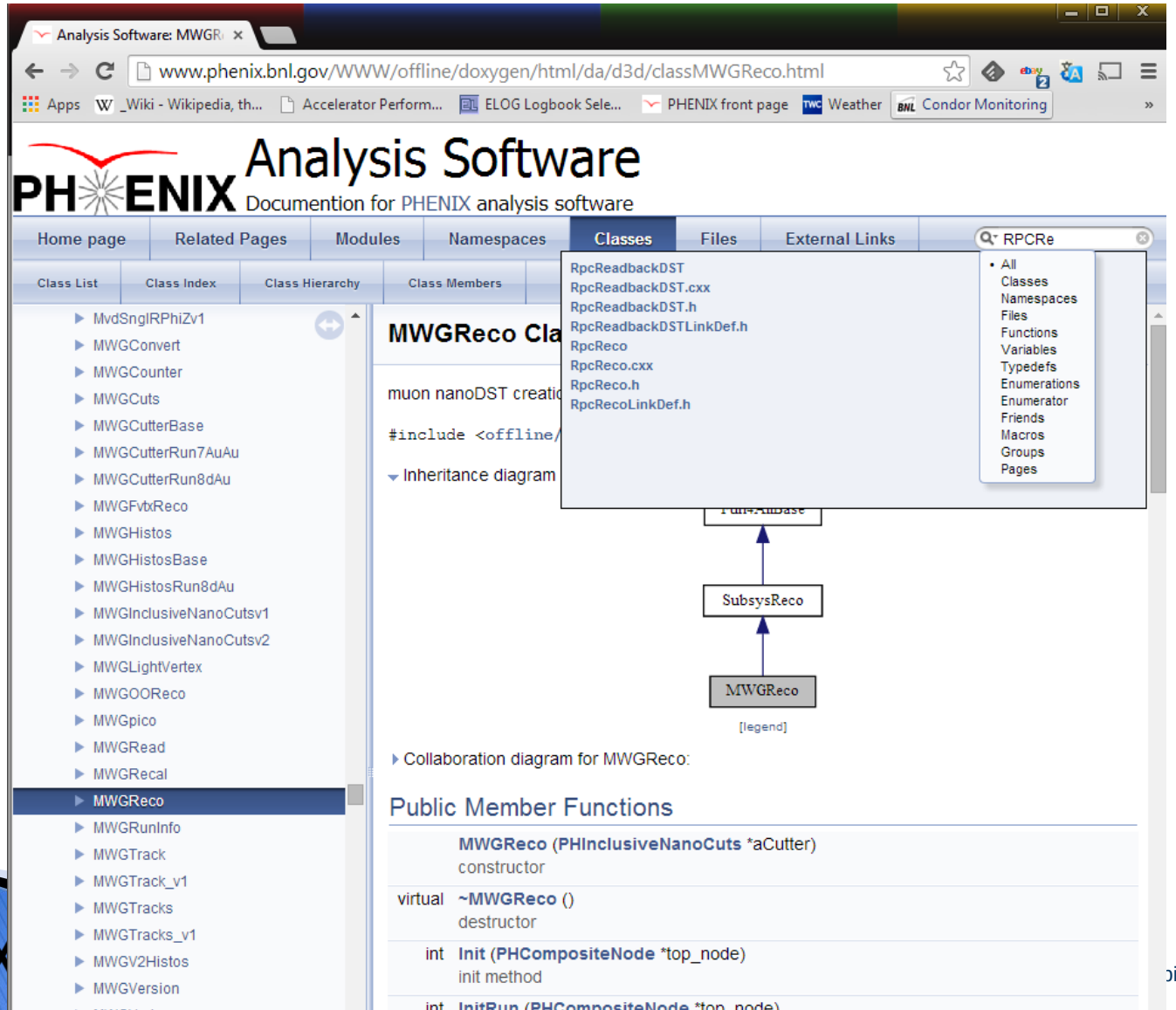
- To ensure your code behave well, sanity checks and pass taxi gatekeeper
- Static code checker:
  - **cppcheck** : Offline Wiki / Cppcheck
  - in your code directory calls
    cppcheck --enable=all *.h *.C
  - Make sure no WARNING or ERROR from your code
- Run-time code checker:
  - **valgrind**: Offline Wiki / Valgrind
  - Run your macro using
    valgrind -v --num-callers=40 --leak-check=full --error-limit=no --log-file=valgrind.log --suppressions=$ROOTSYS/root.supp --leak-resolution=high root.exe 'YourMacro.C(<parameters>)'

Where to find all the code? New PHENIX analysis code documentation site: Doxygen

- http://www.phenix.bnl.gov/WWW/offline/doxygen/html/
  PHENIX Internal -> Computing -> Doxygen
  Auto refresh from CVS

Search and index

- Ever want to know where is one code?



Collaboration/call diagram

- How classes/files are inherited/depended?

PgPostBankWrapper Class Reference

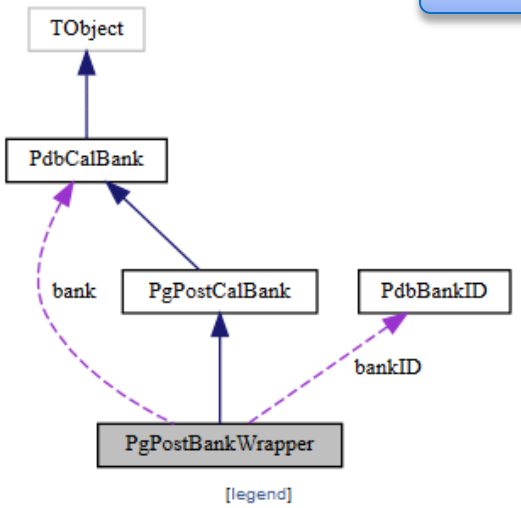
Public Member Functions | Private Member Functions | Private Attributes | List of all members

`#include <offline/database/pdbcal/pg/PgPostBankWrapper.hh>`

Inheritance diagram for PgPostBankWrapper:

Collaboration diagram for PgPostBankWrapper:

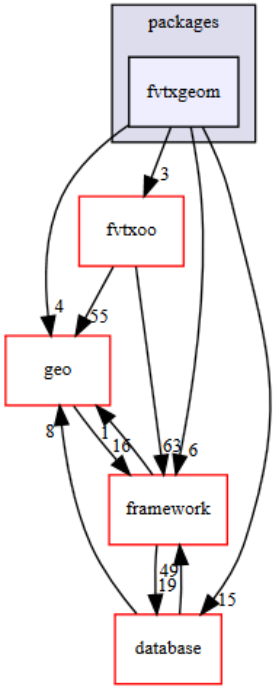
Auto folded



fvtxgeom Directory Reference

View the newest version in PHENIX CVS for this directory "fvtxgeom"

Directory dependency graph for fvtxgeom:



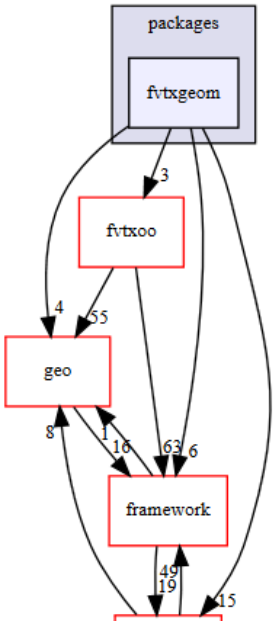
Integration to PHENIX CVS

- Implemented custom copy of doxygen to auto link to PHENIX CVS
- Works on single file or directory

fvtxgeom Directory Reference

View the newest version in PHENIX CVS for this directory "fvtxgeom"

Directory dependency graph for fvtxgeom:



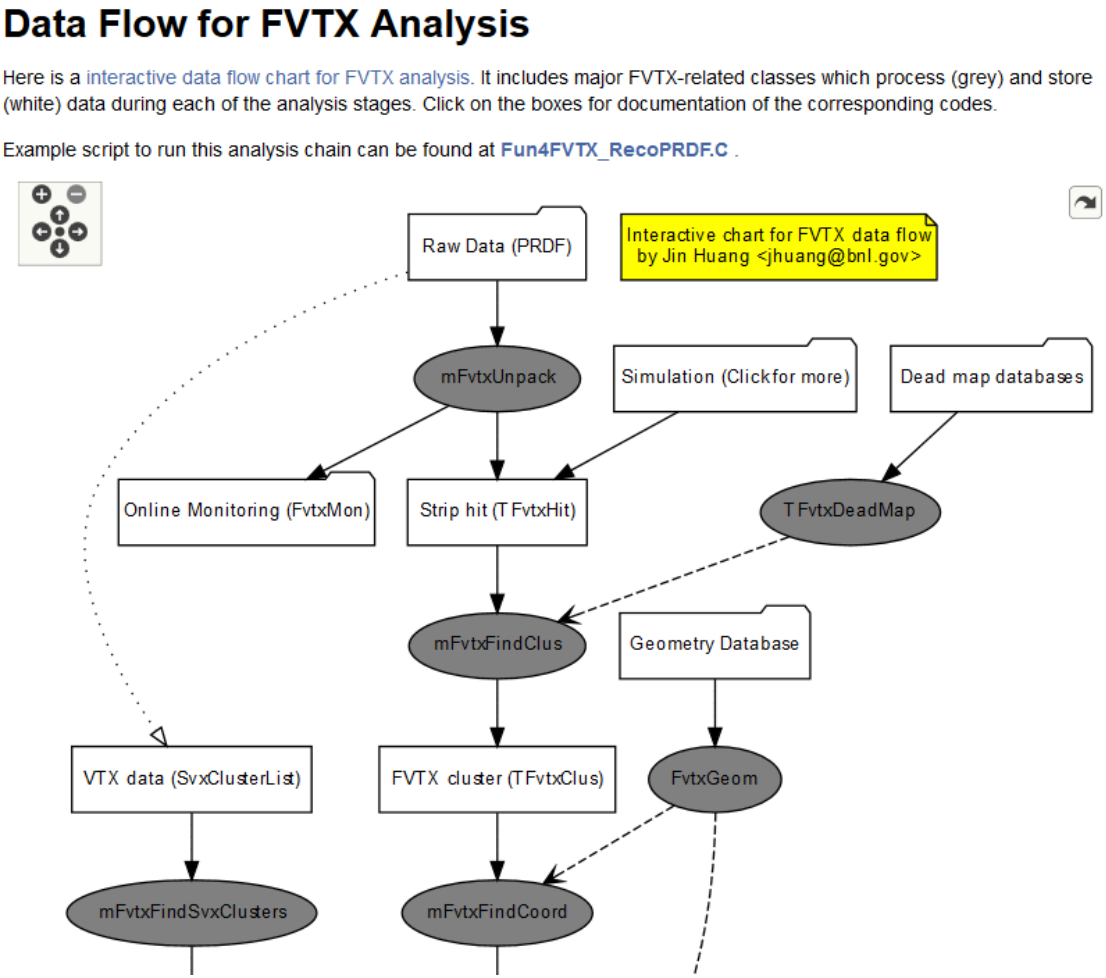
FvtxArm.h

Go to the documentation of this file. Or view the newest version in PHENIX CVS for file FvtxArm.h

```
1  // $Id: FvtxArm.h,v 1.9 2013/11/03 00:34:16 jinhuang Exp $
2
3  #ifndef _FvtxArm_h_
4  #define _FvtxArm_h_
5
15 #include <iostream>
16 #include <stdexcept>
17 #include <vector>
18
19 #include "FVTXGEOM.h"
20 #include "FvtxGeomObject.h"
21 #include "FvtxCage.h"
22
24
27 class FvtxArm : public FvtxGeomObject
28 {
29
30 public:
31
33   FvtxArm(const TFvtxIndex& index) :
34       FvtxGeomObject(index, true), _cages(FVTXGEOM::NumberOfCages, 0)
35   {
36   }
37
39   virtual
```

Data flow chart

- Can be build based on particular production macro
- E.g. FVTX data flow:
- You are welcome to build yours! Welcome to contacting me to start



Comments and suggestions are also welcomed!

Next lecture

- Fun4All based spin analyzer